#### EXPERIMENT – 3.3

**Mapped Course Outcome**

CO1: Identify and describe soft computing techniques and their roles in building intelligent machines.

**AIM:**  
Write a program to implement a Recurrent Neural Network (RNN).

**Theory**  
Recurrent Neural Networks (RNNs) are a class of neural networks that are powerful for modeling sequence data such as time series or natural language. Unlike traditional feedforward neural networks, RNNs have connections that form directed cycles, allowing information to persist. This characteristic makes them suitable for tasks where context and sequential information are crucial.

**Procedure:**

**Step 1: Setup and Installation**

1. **Install Anaconda:**
   * Follow the same installation steps as provided in EXPERIMENT – 1.1.
2. **Install Required Libraries:**
   * Open Anaconda Navigator.
   * Ensure that tensorflow, keras, numpy, matplotlib, and pandas are installed.

**Step 2: Implementing the RNN**

1. **Import Necessary Libraries:**

python

Copy code

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import SimpleRNN, Dense

from tensorflow.keras.datasets import imdb

from tensorflow.keras.preprocessing import sequence

1. **Load and Preprocess the Dataset:**

Use the IMDB dataset, which contains movie reviews for sentiment analysis.

python

Copy code

max\_features = 10000 # Number of words to consider as features

max\_len = 500 # Cut texts after this number of words

batch\_size = 32

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=max\_features)

x\_train = sequence.pad\_sequences(x\_train, maxlen=max\_len)

x\_test = sequence.pad\_sequences(x\_test, maxlen=max\_len)

1. **Build the RNN Model:**

python

Copy code

model = Sequential()

model.add(SimpleRNN(32, input\_shape=(max\_len, 1)))

model.add(Dense(1, activation='sigmoid'))

1. **Compile the Model:**

python

Copy code

model.compile(optimizer='rmsprop', loss='binary\_crossentropy', metrics=['accuracy'])

1. **Train the Model:**

python

Copy code

model.fit(x\_train, y\_train, epochs=10, batch\_size=batch\_size, validation\_split=0.2)

1. **Evaluate the Model:**

python

Copy code

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print(f'Test Loss: {test\_loss}, Test Accuracy: {test\_acc}')

**Step 3: Running the Program**

1. Open Jupyter Notebook from Anaconda Navigator.
2. Create a new Python 3 notebook.
3. Copy and paste the above code sections into the notebook cells.
4. Execute each cell sequentially to build, train, and evaluate the RNN.

**Video Tutorial**

[Recurrent Neural Networks with Keras](https://www.youtube.com/watch?v=G6ogBgJioWk)

**Further Reading**

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2016). Introduction to Anaconda and Python: Installation and setup. Python for research in psychology, 16(5), S5-S11.

**Prospective Viva Questions**

1. Define Recurrent Neural Networks and their primary use.
2. Explain the difference between RNN and traditional feedforward neural networks.
3. Discuss the role of the hidden state in RNNs.
4. Describe how RNNs handle sequential data.
5. Provide examples of real-world applications where RNNs can be effectively used.